Verlustfreie JPEG Drehung – Hinter den Kulissen

Seit 1998 steht mit Erscheinen der Independent JPEG Group Software Version 6b die praktische Möglichkeit zur Verfügung, JPEG Bilddateien verlustfrei in Schritten von 90 Grad zu drehen. Im Zuge der umfangreichen Anwendung von Digitalkameras hat sich dieses Feature in der Praxis weit verbreitet und wurde inzwischen in viele Anwendungen integriert.

Die Funktion der verlustfreien JPEG Drehung lässt sich somit gegenwärtig zwar relativ problemlos nutzen und anwenden, jedoch wissen bisher nur die wenigsten Leute über die Funktionsweise und Hintergründe dieser besonderen Funktion bescheid, und es entstehen immer wieder Fragen und Unklarheiten.

Dieser Artikel soll nun erstmals etwas Licht ins Dunkel der verlustfreien Drehfunktion bringen – es wird hiermit der Versuch unternommen, diese Funktion möglichst anschaulich und ohne Verwendung von mathematischen Formeln zu erklären, so dass einem breiten Publikum Einsicht in die Funktionsweise vermittelt wird und Unklarheiten ausgeräumt werden.

Problem

Das Problem soll anhand folgenden Beispiels[1] dargestellt werden:

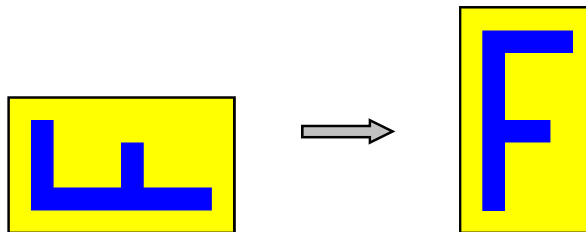


Abb. 1: Drehproblem anhand einer Bilddarstellung des Buchstaben F

Das Bild links entspräche einer Hochformataufnahme mit einer üblichen Querformat-Digitalkamera durch entsprechendes Drehen der Kamera bei der Aufnahme. Im abgespeicherten Querformat-JPEG-Bild liegt der aufgenommene Gegenstand entsprechend auf der Seite, selbst wenn, wie bei einigen Digitalkameras möglich, im EXIF-Header der Datei die tatsächliche Orientierung vermerkt wird.

Durch eine anschließende Drehung des Bildes um 90 Grad im Uhrzeigersinn kann in diesem Fall die tatsächliche Orientierung wieder hergestellt werden.

Lösung des Problems beim Pixelformatbild

Folgende Abbildung zeigt die einfache Lösung des Problems für den Fall, dass das Bild als direktes Pixelformatbild (Pixelmap, Pixelmatrix) vorläge:

[1] Die Idee der Veranschaulichung der Drehproblematik anhand einer Darstellung des Buchstaben F geht zurück auf Adam M. Costello.

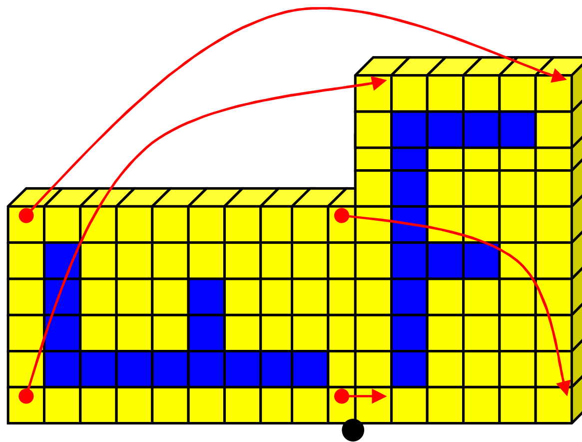


Abb. 2: Einfache Drehung eines Pixelbildes um 90 Grad

Liegt das Bild unmittelbar im Pixelformat vor, so kann eine Drehung um 90 Grad einfach durch geeignete „Umordnung“ der Bildelemente (Pixel) erreicht werden. Dies wird in der Abbildung anhand der 4 Eckpunkte des Bildes demonstriert: Der obere linke Bildpunkt wird im Zielbild an die obere rechte Ecke versetzt, der untere linke Bildpunkt wandert nach oben links, der untere rechte Eckpunkt geht nach unten links, und der obere rechte Bildpunkt wird nach unten rechts platziert. Entsprechend werden alle Bildpunkte der Qelle geeignet ins Ziel versetzt, und es entsteht das gewünschte Resultat. Eine entsprechende programmiertechnische Umsetzung ist recht einfach zu realisieren und in vielen Anwendungen enthalten, die mit Pixelbildern arbeiten.

JPEG Problematik

Die einfache Lösung bei Pixelformatbildern lässt sich leider nicht unmittelbar auf JPEG übertragen, da JPEG Bilder nicht im Pixelformat gespeichert sind. Einen Überblick über den JPEG Prozess gibt die folgende Abbildung:

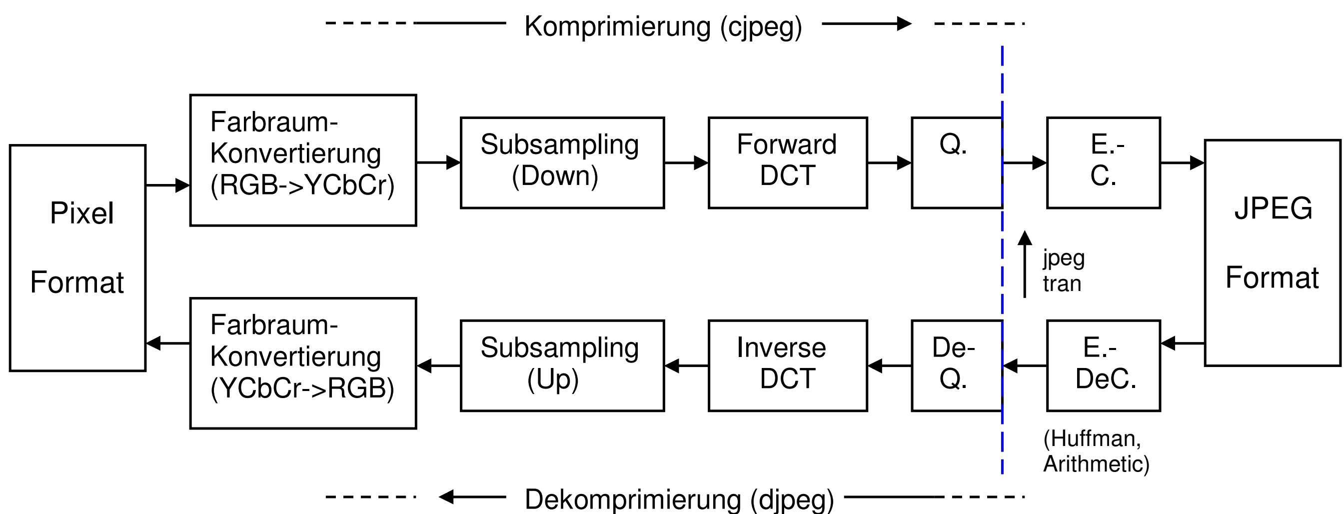


Abb. 3: JPEG Prozess Überblick

JPEG ist bekanntlich ein „verlustbehaftetes“ Format: Bei der Komprimierung werden bestimmte Verluste in Kauf genommen, um die Vorteile wie geringen Speicherbedarf zu ermöglichen.
Würden wir nun unser vorliegendes JPEG Bild komplett in das Pixelformat dekomprimieren, dann nach obigem Schema drehen und anschließend erneut als JPEG komprimieren, so treten dabei unnötige Verluste auf. Im ungünstigsten Fall kann sich der Speicherbedarf sogar deutlich erhöhen, nämlich wenn der Komprimierprozess mit verlustärmeren Parametern gesteuert wird als ursprünglich beim Erzeugen des Ausgangsbildes verwendet, wobei im Endeffekt auf diesem Wege trotzdem immer ein Informationsverlust zu verzeichnen ist.

Diese Vorgehensweise der vollständigen Dekomprimierung und anschließenden Komprimierung sollte daher nach Möglichkeit vermieden und umgangen werden.

Die einzelnen Elemente der JPEG-Prozesskette weisen ein unterschiedliches „Verlustpotenzial“ auf, welches wir uns zunächst klar machen müssen.

Der Teilprozess der Entropie-Kodierung bzw. –Dekodierung am einen Ende der Prozesskette operiert vollkommen verlustfrei, während alle anderen Teile einem mehr oder weniger großen Verlustpotenzial unterliegen. Hier greift deswegen auch die Funktionalität des *jpegtran*-Tools und speziell die verlustfreie Drehfunktion an, wie noch zu demonstrieren ist. Das „Format“, in dem die Bilddaten intern vor der Entropie-Kodierung bzw. nach der Entropie-Dekodierung vorliegen, kann als „Matrix quantisierter DCT-Koeffizienten“ bezeichnet werden. Dies ist für die verlustfreie Drehfunktion wichtig zu wissen, denn hier greift diese an. Damit kann an dieser Stelle schon ein häufig geäußertes Missverständniss aufgeklärt werden: Da die verlustfreie Drehfunktion an der „Matrix quantisierter DCT-Koeffizienten“ ansetzt, und *nicht* unmittelbar am komprimierten Datenstrom, ist in jedem Falle eine erneute Entropie-Kodierung fällig, die den eigentlichen komprimierten JPEG Datenstrom neu erzeugt. Dadurch ändert sich auch die Dateigröße einer verlustfrei gedrehten Datei geringfügig, weil der Entropiekodierung unterschiedliche Eingangsdaten („Symbole“) zugeführt werden.

Der Hauptangriffspunkt zur „Verluststeuerung“ bei JPEG liegt im Teilprozess der Quantisierung, hierbei wird sozusagen „mutwillig“ durch Vorgabe der Verlustgrad bestimmt. Ein weiterer Angriffspunkt zur JPEG Verluststeuerung, der in praktischen Anwendungen leider oft unterschätzt oder übersehen wird, liegt im Teilprozess der Farbunterabtastung (Color Subsampling).

Das eigentliche „Herz“ oder Zentrum des JPEG Prozesses stellt die Diskrete Cosinus-Transformation (DCT oder FDCT [Forward DCT]) bzw. ihr Gegenstück, die Inverse Diskrete Cosinus-Transformation (IDCT), dar. Hierbei werden Pixelwerte in „Schwingungswerte“ umgewandelt und umgekehrt. Diese Umwandlung geschieht „theoretisch“ verlustfrei, die Verlustfreiheit kann jedoch praktisch aufgrund begrenzter Rechengenauigkeit nicht hundertprozentig garantiert werden. Je nach Qualität und Anforderung der Implementierung sind hier mehr oder weniger geringfügige Abweichungen möglich. Ähnliches gilt für den Teilprozess der Farbraumkonvertierung.

Alle diese möglichen Verlustquellen, sowohl die vorgeblich gesteuerten (Quantisierung und Subsampling) als auch die inherenten (DCT/IDCT, Farbkonvertierung), werden bei den verlustfreien *jpegtran*-Operationen und speziell also bei den verlustfreien Drehfunktionen umgangen!

JPEG Schwingungsbilder

Wie bereits erwähnt, stellt die Diskrete Cosinus-Transformation (DCT) das Herz von JPEG dar, und sie verwandelt Pixelwerte in Schwingungswerte, die dann letztendlich in der JPEG Datei gespeichert werden.
Die Schwingungswerte repräsentieren bestimmte Schwingungsmuster, die wir uns im folgenden veranschaulichen wollen.

Zunächst muss noch erwähnt werden, dass die Transformation bei JPEG immer nur auf Blöcke von 8 mal 8 Pixeln angewendet wird, um den Vorgang übersichtlich zu halten und den Aufwand zu

beschränken. Dazu wird das ganze Bild in eine lückenlose Aneinanderreihung solcher 8x8 Pixelblöcke unterteilt, und die Transformation wird dann getrennt auf jeden Block separat angewendet.
Die folgende Abbildung veranschaulicht die JPEG Blockunterteilung:

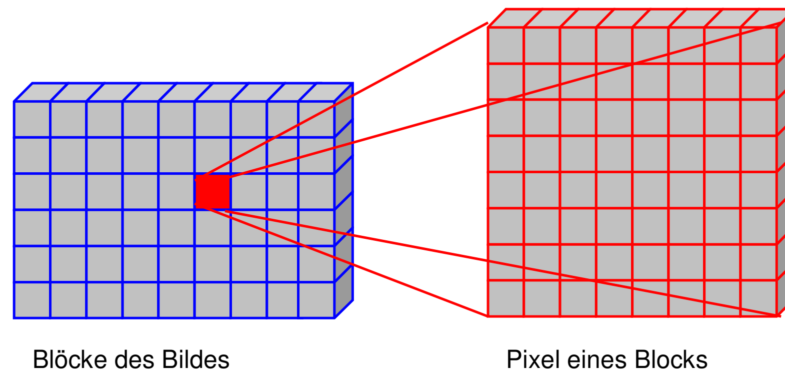


Abb. 4: JPEG Blockbildung

So ein 8x8 Pixelblock stellt nun bereits ein handliches Objekt für die JPEG Transformation dar. Allerdings handelt es sich immer noch um ein zweidimensionales Gebilde, was die Sache technisch etwas erschwert. Um eine weitere Vereinfachung zu erreichen, wird die Transformation zunächst mit einer eindmensionalen 8-Pixel-Folge beschrieben: Aus 8 linearen Pixeln werden 8 lineare Schwingungsmuster gebildet. Anschließend setzen wir eine zweidimensionale Transformation auf eine 8x8 Pixelmatrix dadurch zusammen, dass wir zunächst in einer Richtung 8 mal die lineare Transformation anwenden, und dann in der zweiten Richtung nochmals. Es erweist sich, dass die Reihenfolge, ob erst horizontal (zeilenweise) und dann vertikal (spaltenweise) transformiert wird oder umgekehrt, gleichgültig ist, d.h. es ergibt sich immer das gleiche Resultat. Eine zweidimensionale Transformation, die auf diese Weise aus linearen Transformationen zusammengesetzt ist, nennt man auch „separabel".

Die folgende Abbildung zeigt nun die 8 Schwingungsmuster (auch „Basisfunktionen" genannt) der linearen JPEG DCT:

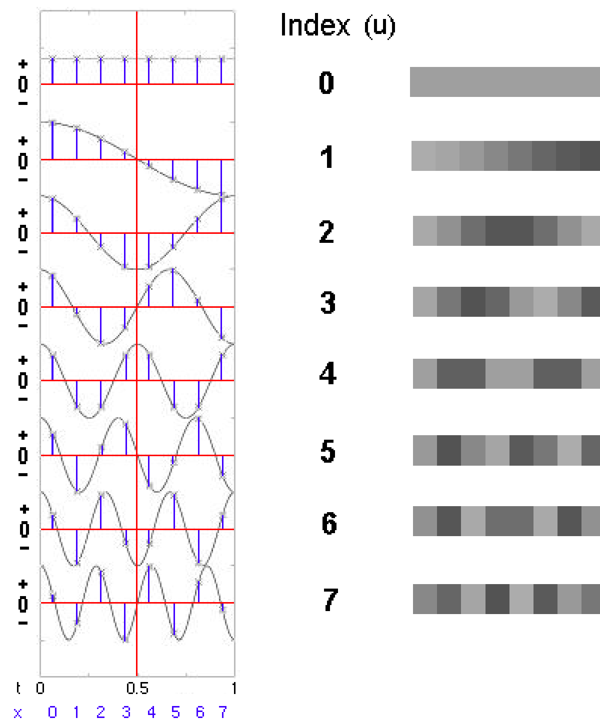


Abb. 5: Schwingungsmuster (Basisfunktionen) der JPEG 8-Punkt DCT

Wir erkennen in den Diagrammen auf der linken Seite die kontinuierlichen (stetigen) Cosinuswellen als Grundgerüst, wovon jeweils 8 diskrete Stützpunkte genommen werden.
Auf der rechten Seite sehen wir die entsprechende Darstellung als Hell-Dunkel-Intensitäten.

Was bedeutet nun die Transformation?
Die Transformation bedeutet, dass wir eine beliebige 8-Punkt-Folge als Kombination solcher Schwingungsmuster repräsentieren können. Darin liegt das eigentliche „Mysterium“ der JPEG DCT: es kann jeder 8-Punkt-Folge eindeutig eine solche Schwingungskombination zugeordnet werden (berechnet mittels DCT Algorithmus), und umgekehrt kann aus der Schwingungskombination wieder eindeutig die ursprüngliche Pixelfolge bestimmt werden (errechnet mittels IDCT).
Die Schwingungswerte, die in der JPEG-Datei gespeichert werden, stellen den Anteil (das „Gewicht“) des jeweiligen Schwingungsmusters entsprechend der folgenden Abbildung dar, wenn wir die lineare Transformation gemäß obiger Erläuterung auf den zweidimensionalen Fall erweitern:

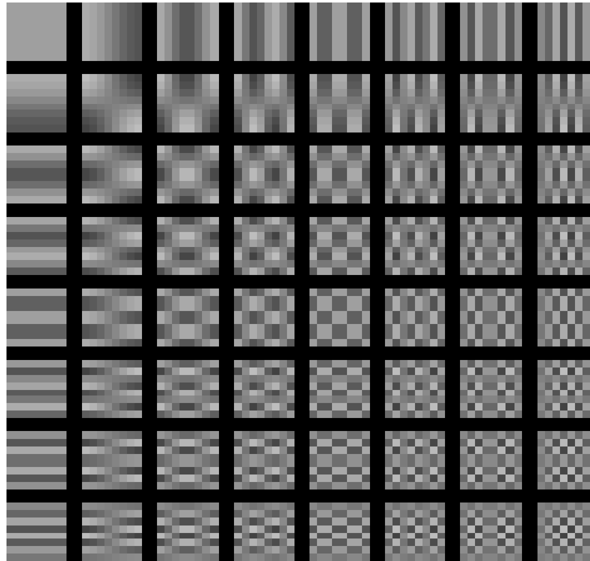


Abb. 6: JPEG DCT 8x8 Schwingungsmuster (Basisfunktionen)

Schrittweise Drehung

Nach all diesen allgemeinen, aber unumgänglichen, Ausführungen erinnern wir uns nun wieder an die eigentliche Aufgabe: Drehung des JPEG Bildes um 90 Grad. Wie bereits gesagt, müssen wir diese Aufgabe im Bereich der DCT-Koeffizienten, also der Schwingungsmuster, lösen, um Verluste zu vermeiden.

Die erste Idee, wenn man sich die erste (1) und die letzte Abbildung (6) ansieht, besteht darin, dass die Schwingungsmuster eine bestimmte Symmetrie bezüglich der Zeilen (horizontal) und Spalten (vertikal) aufweisen, was aufgrund der oben erwähnten separablen Konstruktionseigenschaft auch unmittelbar einleuchtet: Ein Schwingungsmuster einer bestimmten Zeile und Spalte ist spiegelsymmetrisch zu dem entsprechenden Schwingungsmuster mit vertauschter Zeile und Spalte, und zwar bezüglich der Diagonalen (von links oben nach rechts unten, auch Hauptdiagonale genannt).
Daraus ergibt sich: Wenn wir alle Schwingungsmuster (= Koeffizienten in der DCT-Matrix) bezüglich der Hauptdiagonalen spiegelbildlich vertauschen, so werden auch die Pixel des de-transformierten Bildblocks spiegelbildlich bezüglich der Hauptdiagonalen vertauscht. Das spiegelbildliche Vertauschen bezüglich der Hauptdiagonalen einer Matrix nennt man auch „Transponieren“.
Damit haben wir eine relativ einfache Möglichkeit gefunden, ein JPEG Bild anhand der DCT-Koeffizienten zu „transponieren“: Es müssen lediglich die DCT-Koeffizienten aller Blöcke transponiert werden („Intra-Block“) und weiterhin alle Blöcke innerhalb des Bildes („Inter-Block“), also insgesamt lediglich eine etwas umfangreiche Vertauschungsaktion.
Neben den DCT-Koeffizienten müssen auch die Quantisierungswerte, die in Tabellen im Kopf der JPEG Datei gespeichert werden, entsprechend vertauscht werden, denn wir hatten bereits festgestellt, dass wir es mit den *quantisierten* DCT-Koeffizienten zu tun haben.

Das „Transponieren“ stellt nun in der Tat eine wichtige Basisoperation innerhalb der verlustfreien Transformationsfamilie dar, aber leider ist sie erst „die halbe Miete“ auf dem Weg zur Lösung unserer Aufgabe der verlustfreien 90-Grad-Drehung.
Schauen wir uns dazu mal das Resultat der „Transpose“-Operation angewendet auf unser Eingangsbild an:

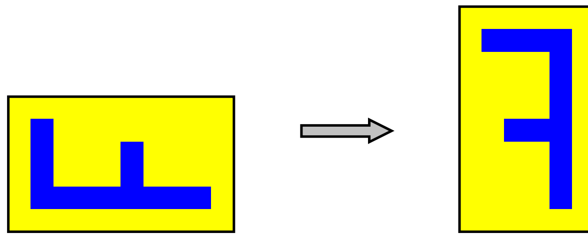


Abb. 7: F-Bild transponiert

Was fehlt uns nun noch auf dem Weg zum gewünschten Dreh-Resultat (vgl. auch Abb. 1)?
Genau, wie unschwer zu erkennen ist, ist das Resultat der „Transpose“-Operation im Vergleich zum korrekt gedrehten Bild horizontal gespiegelt. Was wir also noch brauchen, ist eine horizontale Spiegeloperation!

Nun sehen wir uns noch einmal die linearen Schwingungsmuster in Abb. 5 an. Wir stellen fest, dass abwechselnd jedes zweite Muster entweder horizontal in sich symmetrisch oder antisymmetrisch ist. Die Muster mit den Indices 0,2,4 und 6 (gerade) sind symmetrisch, diejenigen mit den Indices 1,3,5 und 7 sind antisymmetrisch. Daraus ergibt sich ein einfaches Verfahren zur horizontalen Spiegelung: Die Schwingungsmuster mit geraden Indices bleiben unverändert, und bei denen mit ungeraden Indices vertauschen wir das Vorzeichen! Denn, wie man sofort anhand der Diagramme sieht, ist ein Vertauschen des Vorzeichens (Spiegelung an der Mittelachse mit Wert 0) bzw. Invertierung der Hell-Dunkel-Intensität dabei gleichbedeutend mit horizontaler Spiegelung!

Ein entsprechendes Ergebnis lässt sich auch leicht der zweidimensionalen Matrixdarstellung in Abb.6 entnehmen: Um eine horizontale Spiegelung zu erreichen, müssen wir alle Elemente (= DCT-Koeffizienten) mit geradem Spaltenindex unverändert übernehmen (symmetrisch) und bei allen mit ungeradem Spaltenindex das Vorzeichen ändern (antisymmetrisch).

Analog können wir auch das Vorgehen für vertikale Spiegelung ablesen: Alle Elemente mit geradem Zeilenindex übernehmen (symmetrisch) und bei allen mit ungeradem Zeilenindex das Vorzeichen ändern (antisymmetrisch).

Damit sind die „Intra-Block“-Vorgänge für das horizontale/vertikale Spiegeln beschrieben, „Inter-Block“ müssen noch entsprechende einfache Blockvertauschungen vorgenommen werden.

Mit den so gefundenen Grundoperationen können nun durch geeignete Hintereinanderausführung alle anderen Funktionen (Drehungen und Spiegelungen) der verlustfreien Transformationsgruppe ausgeführt werden! Es genügen theoretisch bereits die beiden Grundoperationen Transponieren und Horizontal Spiegeln, daraus lassen sich alle anderen Funktionen zusammensetzen! (Beispielsweise ergäbe sich vertikales Spiegeln auch aus der Folge Transponieren-HorizontalSpiegeln-Transponieren, denn Transponieren bewirkt Zeilen/Spalten-Vertauschen und somit kann vertikales Spiegeln [Zeilentausch] auf horizontales Spiegeln [Spaltentausch] zurückgeführt werden.)
Wie die einzelnen Transformationsfunktionen aus den Grundoperationen zusammengesetzt werden, kann der geneigte Leser den Kommentaren im entsprechenden IJG-Quellcode entnehmen.
Er wird nun anhand dieser Erläuterungen den Code besser nachvollziehen können.

Zusammenfassend kann festgestellt werden, dass sich die gewünschten Drehungen also tatsächlich durch bloße Vertauschungen und Vorzeichen-Änderungen im DCT-Bereich durchführen lassen, und diese Operationen sind zusammen mit der noch erforderlichen Entropie-Kodierung vollkommen verlustfrei, d.h. identisch umkehrbar.

Die folgende Abbildung illustriert nochmal zur Verdeutlichung die beschriebenen Operationen an den Koeffizienten eines 8x8 DCT-Blocks und ihre jeweiligen Wirkungen an jedem der 8x8 Basismuster:

Bild transponiert

Bild horizontal gespiegelt

Bild vertikal gespiegelt

Abb. 8: 2D-Grund-Operationen Transponieren, Horizontal und Vertikal Spiegeln

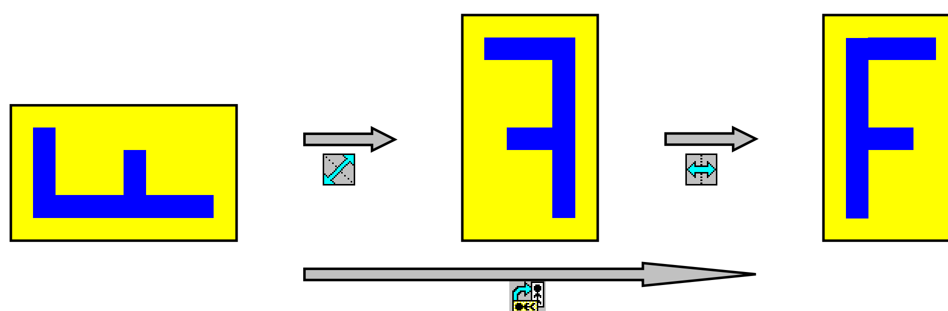


Abb. 9: 90-Grad-Drehung = Transponieren + Horizontal Spiegeln

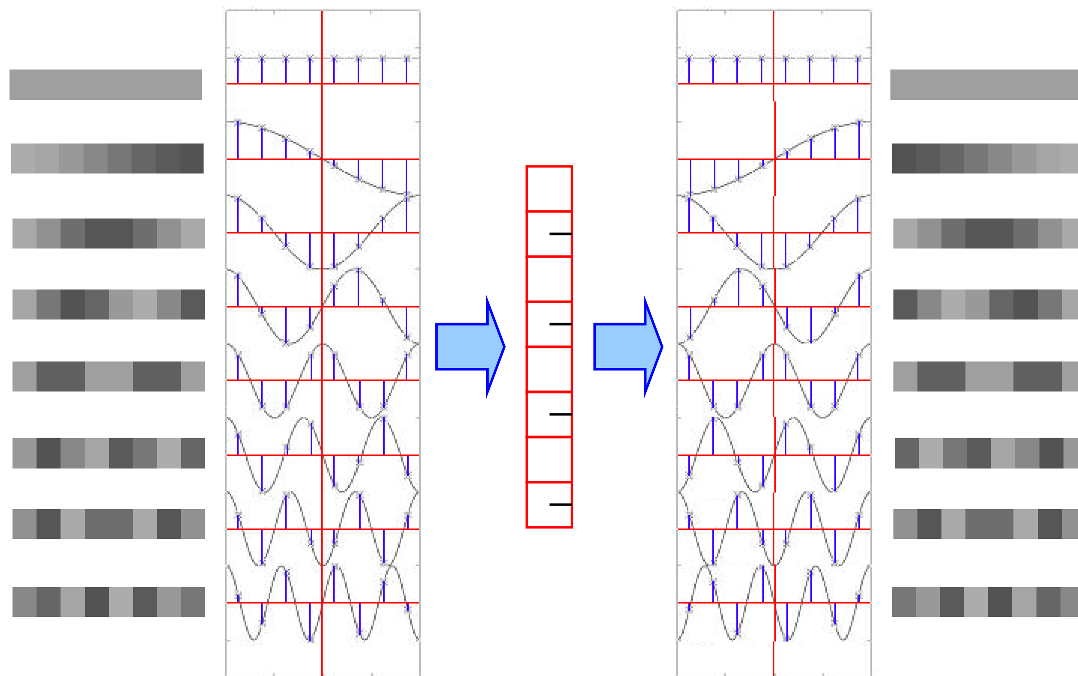


Abb. 10: 1D-Grund-Operation Horizontal Spiegeln

Mathematische Definitionen der FDCT und IDCT

Bisher haben wir es vermieden, die exakten mathematischen Definitionen der FDCT und IDCT anzugeben, weil sie für die Herleitung der verlustfreien Drehfunktion entbehrlich sind und erfahrungsgemäß mathematisch weniger orientierte Leser eher abschrecken würden.
Für die Herleitung der verlustfreien Drehfunktion genügen völlig die in den Abbildungen 5 und 6 gegebenen bildlichen Darstellungen.

Der Vollständigkeit halber seien hier noch die exakten mathematischen Definitionen angegeben. Man betrachte dazu auch Abbildung 5. Ausgangspunkt sind kontinuierliche Cosinus-Schwingungen mit unterschiedlicher Frequenz:

$$\cos tu\pi \quad ; \quad u = 0, 1, ..., 7 \quad ; \quad 0 < t < 1$$

u ist der diskrete Index mit wachsender Frequenz, t ist die kontinuierliche Variable im Interval 0 bis 1. So ergibt sich für $u = 0$ eine konstante Funktion, für $u = 1$ ergibt sich eine halbe Cosinus-Vollschwingung, für $u = 2$ eine ganze Cosinus-Vollschwingung usw.
Die kontinuierliche Variable t wird nun statt im ganzen Interval 0 bis 1 an den diskreten Stellen

$$t = \frac{2x+1}{16} \quad ; \quad x = 0, 1, ..., 7$$

abgetastet. So entsteht innerhalb des Intervals (Block bei JPEG) eine äquidistante Abtastung mit Abstand 1/8 zwischen den Punkten, an den Intervalgrenzen (Blockgrenzen bei JPEG) vorn und hinten ein Abstand von 1/16 zum Randpunkt. Auf diese Weise entsteht durch die Aneinanderreihung von Blöcken bei JPEG ein gleichmäßiger Abstand (1/8) zwischen den Pixeln über das gesamte Bild.
Mit diesen Vorbetrachtungen lassen sich die nun folgenden eigentlichen Definitionen leichter verstehen:

Eindimensionale 8-Punkt FDCT:

$$S(u) = \frac{C(u)}{2} \sum_{x=0}^{7} s(x) \cos \frac{(2x+1)u\pi}{16} \quad ; \quad u = 0, 1, ..., 7$$

Eindimensionale 8-Punkt IDCT:

$$s(x) = \sum_{u=0}^{7} \frac{C(u)}{2} S(u) \cos \frac{(2x+1)u\pi}{16} \quad ; \quad x = 0, 1, ..., 7$$

wobei

$$C(u) = \frac{1}{\sqrt{2}} \quad \textit{für} \quad u = 0$$
$$C(u) = 1 \quad \textit{für} \quad u > 0$$
$$s(x) = 1 - D \ \textit{Pixelwert}$$
$$S(u) = 1 - D \ \textit{DCT} - \textit{Koeffizient}$$

Zweidimensionale 8x8 FDCT:

$$S(v,u) = \frac{C(v)}{2} \frac{C(u)}{2} \sum_{y=0}^{7} \sum_{x=0}^{7} s(y,x) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

Zweidimensionale 8x8 IDCT:

$$s(y,x) = \sum_{v=0}^{7} \frac{C(v)}{2} \sum_{u=0}^{7} \frac{C(u)}{2} S(v,u) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

wobei

$$C(u) = \frac{1}{\sqrt{2}} \quad \textit{für} \quad u = 0$$
$$C(u) = 1 \quad \textit{für} \quad u > 0$$
$$C(v) = \frac{1}{\sqrt{2}} \quad \textit{für} \quad v = 0$$
$$C(v) = 1 \quad \textit{für} \quad v > 0$$
$$s(y,x) = 2 - D \ \textit{Pixelwert}$$
$$S(v,u) = 2 - D \ \textit{DCT} - \textit{Koeffizient}$$